ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

PREGLED MOGUĆNOSTI HTML5 JEZIKA

– Diplomski rad –

Kandidat:

Milica Vukajlović 2009/0464

Mentor:

doc. dr Zoran Čiča

Beograd, April 2014.

SADRŽAJ

SADRŽAJ **2**

SPISAK KORIŠĆENIH SKRAĆENICA **3**

1. UVOD **4**

2. ISTORIJAT HTML JEZIKA **5**

3. PROMENE U ODNOSU NA PRETHODNE VERZIJE **7**

3.1. NOVI STRUKTURNI I SEMANTIČKI ELEMENTI 8

4. FORME **12**

4.1. INPUT TIPOVI „TEL“, „EMAIL“ I „URL“ 12
4.2. INPUT TIPOVI „MONTH“, „DATE“, „NUMBER“ I „RANGE“ 13
4.3. INPUT TIP „SEARCH“ 15

5. MULTIMEDIJA **16**

5.1. VIDEO 16
5.2. AUDIO 18

6. GRAFIKA **19**

6.1. CANVAS 19
6.1.1. Crtanje linije *19*
6.1.2. Crtanje luka *20*
6.1.3. Crtanje izlomljene linije *21*
6.1.4. Tekst na kanvasu *22*
6.1.5. 3D tekst sa senkom *22*
6.1.6. Različiti stilovi unutar trouglova *23*
6.2. SVG (SCALABLE VECTOR GRAPHICS) 25
6.2.1. Krug *25*
6.2.2. Pravougaonik *26*
6.2.3. Elipsa *26*

7. APLIKACIJE **28**

7.1. LOKALNO SKLADIŠTENJE PODATAKA 28
7.1.1. LocalStorage objekat *28*
7.1.2. SessionStorage objekat *30*
7.2. APLIKACIJE SA KEŠIRANJEM PODATAKA 30
7.3. JAVASCRIPT RADNICI (WORKERS) 31
7.4. SERVER – SENT DOGAĐAJI 32
7.5. GEOGRAFSKA LOKACIJA 32

8. ZAKLJUČAK **34**

LITERATURA **35**

SPISAK KORIŠĆENIH SKRAĆENICA

Skraćenica	Pun naziv
CERN	The European Organization for Nuclear Research
IETF	Internet Engineering Task Force
W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
SVG	Scalable Vector Graphics
DOM	Document Object Model

1. UVOD

HTML (HyperText Markup Language) je jezik koji služi za prikazivanje veb strana. Cilj ovog rada je predstavljanje najnovije verzije pomenutog jezika, **HTML5**.

HTML5 u odnosu na prethodne verzije donosi mnogo noviteta i mogućnosti. Neki delovi koda su dosta jednostavniji, a izgled stranice bogatiji. Moguće je uključiti multimedijalni sadržaj bez korišćenja dodataka, moguć je rad sa aplikacijama, grafikom,... Međutim da bi se u potpunosti „iskoristio“ HTML5, potrebno je poznavanje JavaScript-a koji omogućava crtanje na kanvasu, keširanje podataka, određivanje geografske lokacije,..

Nijedan brauzer trenutno nema punu HTML5 podršku, ali većina brauzera nastavlja da dodaje nove karakteristike svojim poslednjim verzijama. U ovom radu, svaki od novih elemenata je testiran na Exploreru, Mozzili, Operi, Chrome i Safari brauzeru.

Rad je podeljen na osam poglavlja. Prvo poglavlje je uvod u kome su date osnovne informacije o radu. U drugom poglavlju je dat istorijski razvoj HTML jezika. Treće poglavlje opisuje promene u odnosu na prethodne verzije i nove elemente. U četvrtom poglavlju su obrađene forme i novi input elementi i dati su neki od primera u kojima mogu da se upotrebe. Peto poglavlje se bavi multimedijom i mogućnošću ubacivanja audio i video fajlova u HTML5 stranicu. Šesto poglavlje ispituje mogućnosti rada sa grafikom, dati su različiti primeri crtanja po kanvasu i opisan je SVG. Sedmo poglavlje opisuje aplikacije i novitete koje uvodi HTML5 na tom polju, dok se osmo tj. poslednje poglavlje bavi prednostima i manama HTML5 jezika.

2. ISTORIJAT HTML JEZIKA

Jedan od najzaslužnijih za razvoj HTML-a je Tim Berners-Lee koji je tokom svog rada u CERN-u predložio prototip sistema za istraživače uz pomoć koga bi mogli da se koriste i dele dokumenti. 1989. godine je napisao rad kojim predlaže hipertekst sistem baziran na Internetu. U drugoj polovini 1990. godine, specificirao je HTML i napisao softver i za server i za brauzer, ali taj projekat nije usvojen od strane CERN-a.

Prvi pisani javni dokument o HTML-u je „HTML Tags“ u kome je opisano 20 elemenata koji su činili relativno jednostavan dizajn. HTML se zasnivao na SGML-u (*Standard Generalized Markup Language)* i definisan je od strane IETF-a radom „Hypertext Markup Language (HTML)“ koji je predlog prve specifikacije HTML-a. Pomenuti rad je uključivao i SGML Document Type Definition koji definiše pravila, tj. gramatiku.

1993. god. Dave Ragett predlaže standardizaciju već implementiranih elemenata kao što su tabele i forme, a 1994. god. Berners-Lee je osnovao W3C koji je usvojen kao standard na kojem će se sve veb strane u budućnosti zasnivati.

Prva HTML specifikacija bio je **HTML 2.0** koji je uključivao ideje iz HTML, ali sa osnovnom namerom da se razlikuje od prethodnih verzija. Zatim nastaje **HTML 3.0** koji se bazira na HTML2.0, ali ta verzija nikada nije implementirana i zamenio ga je **HTML 3.2**. Sledeća verzija standarda je **HTML 4** koji je proširio HTML sa mehanizmima za stil, pisanje, okvire, tekst, bogatije tabele. **HTML 4.01** je verzija koja ispravlja greške i pravi određene izmene (npr. novi stilovi za dokumente koji su bazirani na W3C tehničkim izveštajima).

Nova verzija jezika zvala se **XHTML1.0**. X potiče od reči „extensible”. Sadržaj XHTML je identičan HTML-u 4.01. Nema novih elemenata ili atributa. Jedina razlika je u sintaksi. Dok HTML pruža autorima slobodu da pišu elemente i atribute kako im odgovara, XHTML zahteva od autora da prati pravila XML-a *(Extensible Markup Language)*, jezika sa striktnijim sintaksnim pravilima. Ova pravila omogućavaju autoru da koristi jedinstveni stil pisanja. Dok su u prethodnim verzijama tagovi i atributi mogli da se pišu i velikim i malim slovima i kombinovano, XHTML 1.0 zahteva da se sve piše malim slovima. Prema standardu, XHTML se koristi zajedno sa CSS *(Cascading Style Sheets)* jezikom pomoću koga se olakšava dizajn i ažuriranje velikih veb strana. XHTML zahteva završetak komandi, zahteva da atributi budu pod navodnicima, kao i komandu DOCTYPE u kojoj se specificira varijanta standarda. Elementi u HTML-u se opisuju atributima, a u XHTML-u osobinama. Neki atributi su potisnuti u XHTML-u. **XHTML 1.1** se razvio od striktne verzije XHTML 1.0, ali nije zaživeo. Isto se desilo i sa **XHTML 2.0**, koji je prekinuo kompatibilnost sa HTML standardom. Uporedo sa tim se razvijao i HTML5 gde je posebna pažnja posvećena kompatibilnosti sa aplikacijama.

HTML5 je jezik koji se koristi za struktuiranje i prezentovanje sadržaja za www. To je peta revizija HTML standarda koja se i dalje dopunjuje i razvija. Osnovni ciljevi su poboljšanje jezika i podrška za najnovije multimedijalne sadržaje. Bitno je da brauzer podržava HTML5 bez obzira koji se uređaj koristi. HTML5 je proizvod saradnje W3C i WHATWG. WHATWG je radio na razvoju veb formi i aplikacija, dok je W3C radio na XHTML 2.0. 2006. godine su odlučili da sarađuju i stvore novu verziju HTML-a. Trenutno postoje dve paralelne verzije HTML5, jedna

službena na kojoj radi W3C i neslužbena na kojoj rade stručnjaci iz Apple-a, Mozille, Opere i Google-a, koji imaju veliki uticaj na razvoj Interneta, iako iza sebe nemaju formalnu organizaciju kao što je W3C. Ipak, urednici HTML5 specifikacije za oba tima su isti, što znači da grupe međusobno sarađuju i da ćemo na kraju imati jedan standard za koji se smatra da će biti gotov do kraja ove godine. Urednici su Ian Hickson iz Googlea i David Hyatt koji radi za Apple.

3. PROMENE U ODNOSU NA PRETHODNE VERZIJE

HTML5 uvodi mnoge sintaksne promene. One uključuju nove <video>, <audio> i <canvas> elemente kao i integraciju SVG sadržaja (koji zamenjuju korišćenje <object> tagova). Ove karakteristike su dizajnirane tako da lako uključuju i rukuju multimedijalnim i grafičkim sadržajem na vebu bez potrebe uključivanja plugin-ova i API-ja. Drugi novi elementi, kao što su na primer <section>, <article>, <header> i <nav> su kreirani da obogate semantički sadržaj dokumenta. S druge strane, neki elementi i atributi su uklonjeni, kao, na primer, <font> i koji su prevaziđeni korišćenjem mnogo moćnijeg CSS-a. HTML5 se često poredi sa Flash-om iako su ove dve tehnologije potpuno različite. Obe uključuju karakteristike za puštanje audio i video sadržaja unutar veb strana i za korišćenje SVG komponenti. HTML5 sam po sebi se ne može koristiti za animacije i interaktivnost već samo u kombinaciji sa CSS3 i Javascript-om.

Neke od ideja i principa kojima se vode ljudi koji učestvuju u razvoju HTML5 su:

- Nove karakteristike treba da se baziraju na HTML, CSS, DOM i Javascript-u
- Smanjiti potrebu za eksternim plugin-ovima (Flash)
- Bolja obrada grešaka
- Više tagova koji bi zamenili skripte
- HTML5 bi trebalo da ne zavisi od uređaja na kome se koristi
- Proces razvoja treba da bude dostupan javnosti

Sledeći elementi su korišćeni u HTML4.10, a izbačeni iz HTML5 standarda:

- <acronym>
- <applet>
- <basefont>
- <big>
-
- <dir>
- <font>
- <frame>
- <frameset>
- <noframes>
- <strike>
- <tt>

HTML5 takođe pojednostavljuje kod, tako da se <!DOCTYPE> deklaracija kojom počinje HTML dokument ranije pisala kao:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">.
```

Postojalo je više načina da se DOCTYPE naredba zapiše, u zavisnosti od verzije HTML-a, a u HTML5 se jednostavno piše kao:

```
<!DOCTYPE html> .
```

Umesto meta taga koji je izgledao:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> ,
```

sada imamo:

```
<meta charset="utf-8"> .
```

U ranijim verzijama html tag sa atributima se zapisivao na sledeći način:

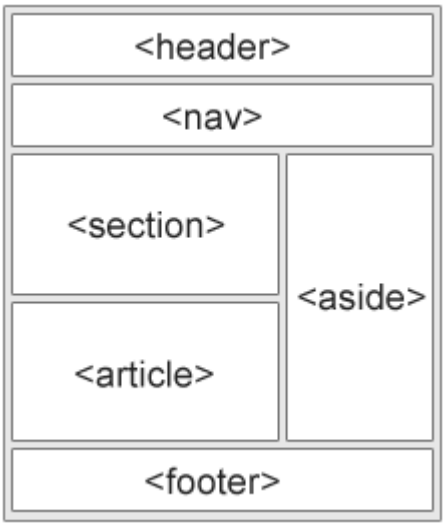
```
<html xmlns=http://www.w3.org/1999/xhtml lang="en" xml:lang="en">,
```

u HTML5:

```
<html lang="en"> .
```

3.1. Novi strukturni i semantički elementi

Struktura jedne HTML5 stranice prikazana je na slici 3.1.1.



Slika 3.1.1. Struktura HTML5 stranice

Slika 3.1.1. prikazuje nove strukturne elemente koji se nalaze na HTML5 stanici. Na samom vrhu HTML5 stranice nalazi se zaglavlje u kome mogu da se nađu informacije o samoj stranici. Zatim sledi set navigacionih linkova koji se definišu <nav> tagom. Tag <section> definiše određene delove stranice u okviru kojih se može naći više različitih članaka koje opisuje tag <article>. Jedan deo stranice koji se nalazi sa strane može da sadrži dodatne informacije, vesti i sličan sadržaj koji se smešta unutar taga <aside>. I na dnu same stranice se nalazi futer gde se nalaze informacije o autoru, autorskim pravima, kontakt podaci i slično.

HTML5 uvodi nove elemente radi boljeg struktuiranja, a neki od njih su dati u tabeli 3.1.1.

Tabela 3.1.1. Novi elementi u strukturi stranice

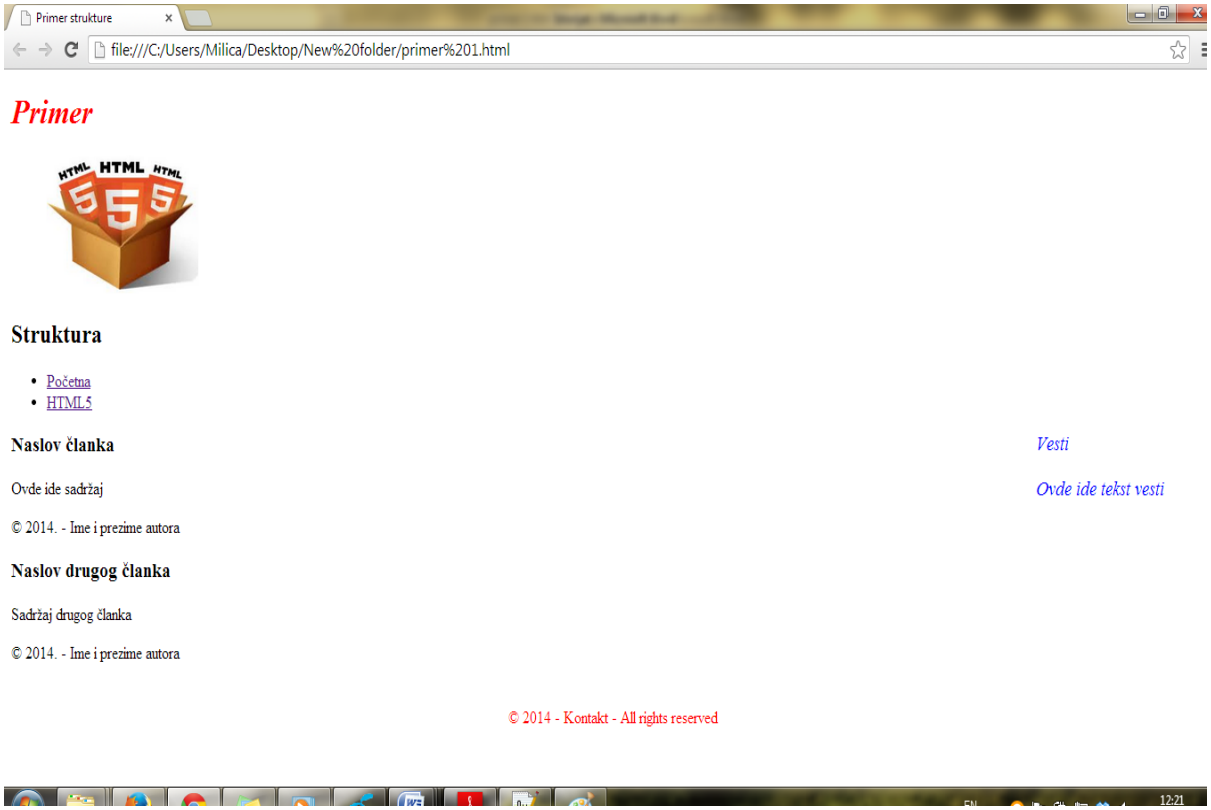
Tag	Opis
<header>	Sadrži informacije o sekciji ili stranici
<nav>	Definiše linkove za navigaciju
<section>	Definiše članke, komentare i sličan sadržaj
<article>	Sličan elementu i definiše određene delove stranice
<aside>	Definiše sadržaj koji se nalazi sa strane u odnosu na sadržaj stranice
<footer>	Definiše futer za dokument ili sekciju, najčešće se tu nalaze podaci o autoru, pravima, itd.
<command>	Definiše komandno dugme koje korisnik može da pozove
<details>	Definiše dodatne detalje koje korisnik može da vidi ili sakrije
<**figure**>	Određuje prostor za sadržaje kao što su ilustracije, dijagrami, slike,..
<mark>	Definiše označen (markiran) tekst
<progress>	Prikazuje napredak zadatka
<hgroup>	Definiše skup od <h1> do <h6> elemenata kada se naslov sastoji iz više nivoa
<meter>	Definiše skalarno merenje unutar poznatog opsega
<wbr>	Definiše mogući prelazak u sledeći red
<**time**>	Definiše datum/vreme

U nastavku se nalazi html kod koji opisuje html strukturu i koristi neke od gore navedenih elemenata.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title> Primer strukture </title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<header>

<hgroup>
<span style="color:red;font-style:italic;">
<h1> Primer </h1> </span>
<figure>
<img src="html5.jpg" width="170" height="128" />
</figure>
<h2> Struktura </h2>
</hgroup>
</header>
<nav>
<ul>
<li> <a href="#" > Početna </a></li>
```

```
<li> <a href="#">  HTML5 </a></li>
</ul>
</nav>
<section>
<article>
<aside style="font-size:larger;font-
style:italic;color:blue;float:right;width:200px;">
 Vesti
<p> Ovde ide tekst vesti </p>
</aside>
<header>
<h1> Naslov članka </h1>
</header>
<p> Ovde ide sadržaj </p>
<footer>
  &copy; 2014. - Ime i prezime autora
</footer>
</article>
<article>
<header>
<h1> Naslov drugog članka </h1>
</header>
<p> Sadržaj drugog članka </p>
<footer>
&copy; 2014. - Ime i prezime autora
</footer>
</article>
</section>
<br> <br>
<footer style="text-align:center; color:red;"> &copy; 2014 -  Kontakt - All
rights reserved </footer>
</body>
</html>
```

Iz primera se može videti da se elementi kao što su <header> i <footer> mogu koristiti više puta u okviru stranice i ponašaju se kao klase. Oni nisu predviđeni da predstavljaju samo vrh i dno stranice, već se koriste da predstave <header> i <footer> svakog dela dokumenta. Stil za HTML5 elemente je mogao da se stavi u CSS, ali bi se u tom slučaju koristio tag i id atribut.



Slika 3.1.2. Izgled stranice kada se pokrene prethodni kod

Slika 3.1.2. predstavlja grafički prikaz prethodnog koda. Vidi se postojanje delova stranice prikazanih na slici 3.1.1. U zaglavlju stranice se nalazi naslov i slika čije su dimenzije podešene u kodu. Zatim slede linkovi koji su predstavljeni u obliku liste. Ispod toga je sekcija u okviru koje se nalaze dva članka, a svaki od njih ima svoj naslov, deo gde ide sadržaj i futer iz koga se može videti ko je autor članka. Na desnoj strani se nalaze vesti čija je pozicija podešena pomoću naredbe *float*. I na dnu same stranice se nalazi futer sa kontakt podacima.

4. FORME

Forme su suštinski deo svake veb aplikacije, njih popunjava korisnik i to je ono što čini veb stranice interaktivnim. HTML5 uvodi nekoliko novih „input" tipova koji omogućavaju bolju kontrolu unosa podataka i validaciju. Jedini problem je što pojedini brauzeri nemaju podršku za sve nove input tipove.

Novi input tipovi su:

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

4.1. Input tipovi „tel", „email" i „url"

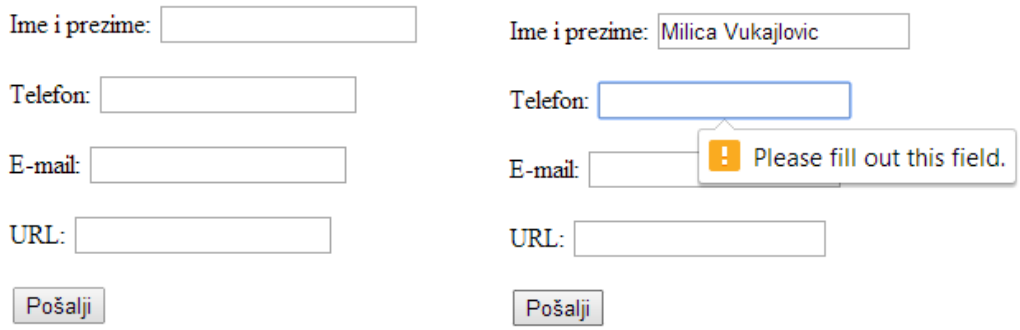
Input tip – *tel* omogućava korisniku da unese telefonski broj, *email* omogućava unos e-mail adrese, a *url* URL adrese.

Sledeći kod daje primer formulara gde su korišćeni neki od novih input tipova.

```
<form>
 <p><label>Ime i prezime: <input type="text" required></label></p>
 <p><label>Telefon: <input type="tel" required></label></p>
 <p><label>E-mail: <input type="email" required></label></p>
 <p><label>URL: <input type="url" required> </label></p>
  <p><button>Pošalji</button></p>
</form>
```

Pored novih input tipova, u kodu se vidi i atribut *required***,** koji dodatno pojednostavljuje kod. Do HTML5, bile su potrebne JavaScript funkcije koje proveravaju da li su sva polja

popunjena, a sada se to jednostavno proverava korišćenjem pomenutog atributa. Problem je što pojedini brauzeri ne podržavaju sve input tipove, tako da input tipove *url* i *email* ne podržava Safari brauzer, dok input tip *tel* nije podržan od Explorera, Opere, Mozzile, Chrome i Safari brauzera.



Slika 4.1.1. Izgled formulara (levo) i izgled formulara kada se popuni prvo polje i klikne na „Pošalji“ (desno)

Slika 4.1.2. Izgled formulara kada se pogrešno ukuca e-mail adresa (levo) i url (desno)

Kada se pokrene prethodni kod, na brauzeru se prikazuje formular kao na slici 4.1.1. (levo). Kada se popuni prvo polje i odmah klikne na „Pošalji“, izaći će obaveštenje da se mora popuniti i prvo sledeće polje zato što je u kodu korišćen atribut *required* koji to zahteva. Sa slike 4.1.2. (levo) se vidi da input tip – *email* upozorava korisnika da nije pravilno uneo e-mail adresu, odnosno da nedostaje znak „@“, a na istoj slici desno izlazi upozorenje da se pravilno unese URL adresa.

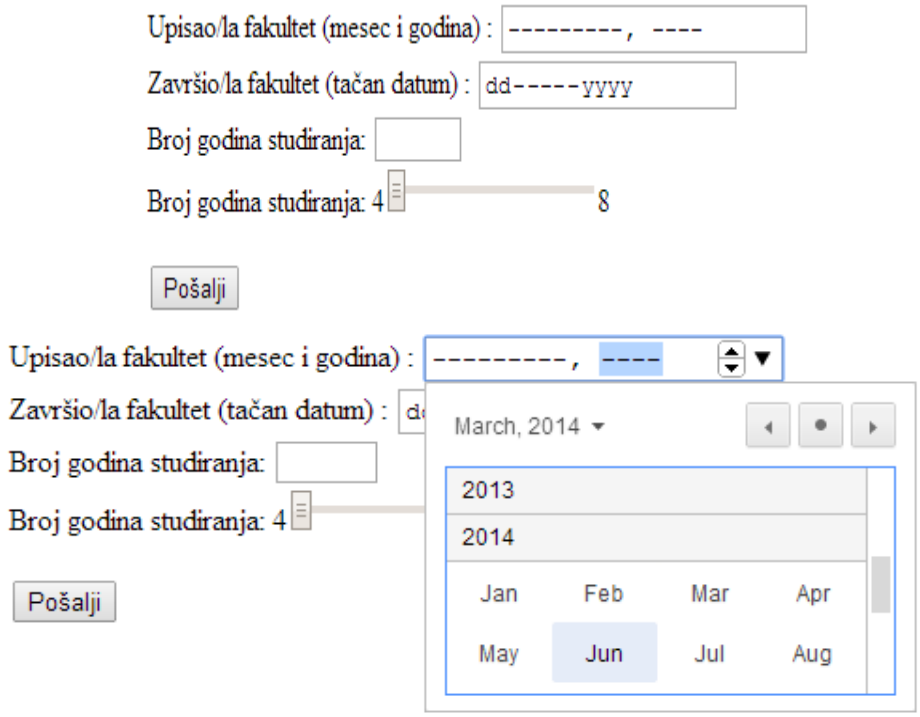
4.2. Input tipovi „month“, „date“, „number“ i „range“

Input tip – *month* omogućava korisniku da izabere mesec i godinu, dok *date* omogućava da se izabere datum. *Number* se koristi kad je potrebno popuniti polja koja sadrže broj, a *range* prikazuje skalu na kojoj se može izabrati broj. *Number* i *range* mogu da se ograniče tako što se odredi minimalna i maksimalna vrednost koristeći atribute *min* i *max*.

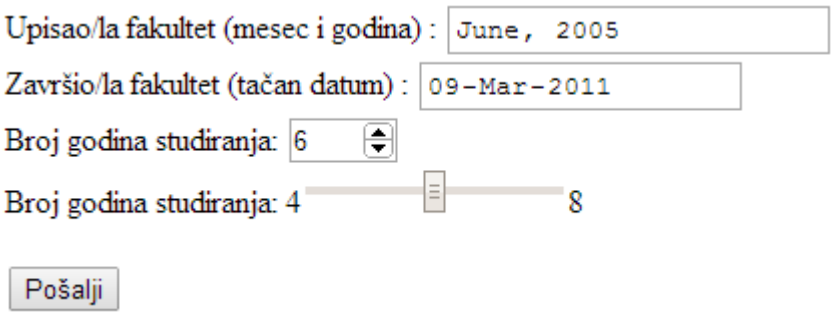
Sledeći primer koristi gore navedene input tipove:

```
Upisao/la fakultet (mesec i godina) : <input type="month" name="upis"><br/>
Završio/la fakultet (tačan datum) : <input type="date" name="zavrsetak">
<br/>
Broj godina studiranja: <input type="number" name="broj" min="4" max="8">
<br/>
Broj godina studiranja: 4<input type="range" name="broj" min="4" max="8">8
<br/><br/>
<button> Pošalji </button>
```

U navedenom primeru su prikazane dve mogućnosti za unos broja godina studiranja. Jedno koristi input tip *number* gde se ručno unosi broj od 4 do 8 ili pomoću strelica koje se nalaze na kraju polja. Drugi način je pomoću input tipa *range* gde se prikazuje skala i prevlačenjem pravougaonika, bira se odgovarajući broj godina studiranja. U kodu, ispred i iza dela gde se definiše input tip *range*, stoje brojevi 4 i 8, respektivno. Oni pokazuju koliki je minimalan odnosno maksimalan broj godina studiranja koje korisnik može da izabere.



Slika 4.2.1. Osnovni izgled (gore), izgled kada se popunjava mesec i godina (dole)



Slika 4.2.2. Popunjen formular

Na slici 4.2.1. (gore) se može videti izgled stranice kada se pokrene prethodni kod. Na istoj slici ispod, kada se popunjava prvo polje, tj. mesec i godina upisa fakulteta, otvara se prozor gde je moguće selektovati željeni mesec i godinu. Izabrani mesec ili godina mogu se pomoću strelica smanjivati ili povećavati za jedan. Na slici 4.2.2. je prikazan izgled stranice kada se popuni formular.

Input tip – *range* je podržan od strane svih brauzera, dok se *date*, *month* i *number* ne prikazuju u Mozzili i Exploreru.

4.3. Input tip „search“

Definiše polje za pretragu nekog sajta ili brauzera, kao što je npr. Google.

Primer:

```
<input type="search" name="pretraga" placeholder="Pretraži sajt"
style="color:blue;"><br/>
<button style="color:blue"> Pretraga </button>
```

Slika 4.3.1. Primena input tipa search

Prethodni kod prikazuje stranicu kao na slici 4.3.1. levo. Unutar polja u koji se kuca tekst, vidi se da piše “Pretraži sajt”. To je omogućeno korišćenjem atributa *placeholder*. Na istoj slici desno je prikazan izgled prethodnog koda kada se ukuca ono što se traži.

Search je podržan samo od Chrome i Safari brauzera, što ga još ne čini pogodnim za korišćenje. Što se tiče pomenutog atributa, podržan je od Mozzile, Opere, Chrome i Safari brauzera, Explorera 10, dok na starijim verzijama Explorera ne radi.

5. MULTIMEDIJA

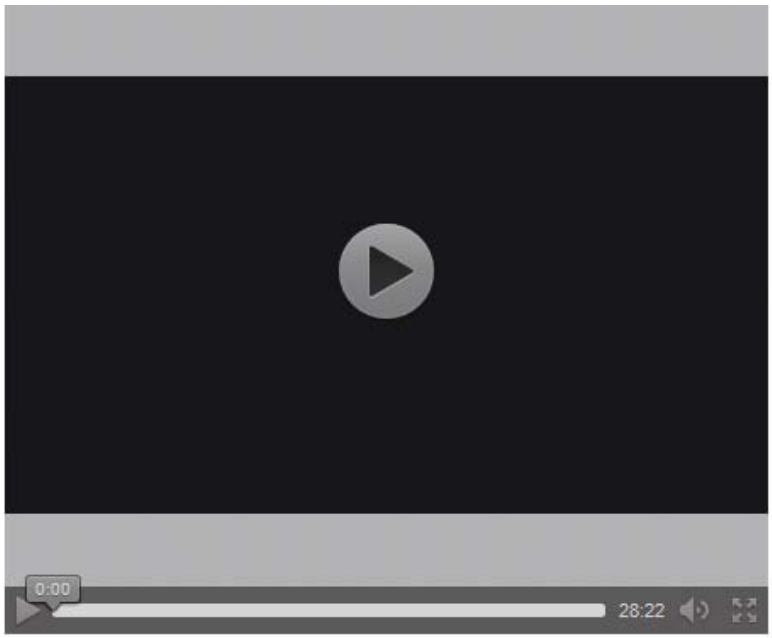
HTML5 uvodi nove elemente za multimedijalni sadržaj. Omogućava puštanje video snimaka ili filmova i audio sadržaja.

5.1. Video

Element <*video*> omogućava prikazivanje video snimka na veb strani. Pre HTML5, to je bilo moguće samo korišćenjem raznih dodataka (plug-ins), Apple QuickTime ili Adobe Flash. Najveći problem je u formatu videa i za sada se mogu prikazivati sadržaji u mp4, ogg ili webM formatu, dok je ostale potrebno konvertovati u neki od tih formata.

Sledi primer upotrebe <*video*> elementa:

```
<video width="450" height="400" controls>

<source src="Cardio.mp4" type="video/mp4">

Ovaj brauzer ne podržava video element.
</video>
```



Slika 5.1.1. Izgled stranice kada se pokrene kod sa video elementom

Slika 5.1.2. Izgled stranice kada se pokrene video

U kodu, pomoću atributa *width* i *height* se definiše veličina videa kako bi prilikom učitavanja brauzer zauzeo odgovarajući prostor za njega. Atribut *control* omogućava kontrolu nad videom, pa video može da se pauzira, pokrene, proširi na čitav ekran, podesi jačina zvuka što se može videti na slikama 5.1.1. i 5.1.2.. Da se nije koristio atribut *control* deo stranice predviđen za video bi bio kao na slici 5.1.3., odnosno crn ekran. Poželjno je u okviru video taga staviti i poruku obaveštenja, u slučaju da brauzer ne podržava video tag.

Slika 5.1.3. Izgled stranice bez atributa control

Različiti formati videa su podržani od strane različitih brauzera. Mp4 format ne podržava Explorer 8 i starije verzije i Opera, dok video u webM i ogg formatu ne podržavaju Explorer i Safari.

5.2. Audio

HTML5 je uveo audio element koji služi za puštanje zvuka, bez potrebe korišćenja flash-a ili nekog drugog plugin-a. Trenutno, postoje tri podržana formata: mp3, ogg i wav.

Primer upotrebe <*audio*> elementa:

```
<audio  controls>
  <source src="anybody.mp3" type="audio/mpeg">
Brauzer ne podržava audio element.
</audio>
```

Slika 5.2.1. Izgled stranice kad se pusti audio snimak

Kao i kod video elementa, koristi se atribut *control* koji omogućava puštanje audio snimka, pauziranje i podešavanje jačine zvuka, što se može videti i na slici 5.2.1..

Mp3 format audio elementa ne podržava Opera, wav format Explorer, a ogg format ne podržavaju Explorer i Safari.

6. GRAFIKA

Do HTML5, animacije i vizuelne efekte je bilo moguće realizovati samo korišćenjem CSS-a i JavaScript-a. HTML5 donosi nove mogućnosti i uvodi elemente koje je moguće koristiti za grafiku na vebu:

- Canvas
- SVG (Scalable Vector Graphics)

6.1. Canvas

Canvas element se koristi za crtanje grafike, u hodu, preko skriptovanja (najčešće preko JavaScript-a). Canvas sadrži nekoliko metoda za crtanje krugova, putanja, karaktera, ispisivanje teksta i dodavanje slika.

Kanvas je pravougaona regija na HTML stranici koja se definiše preko <*canvas*> elementa. Nema ni okvir ni sadržaj. Definiše se na sledeći način:

```
<canvas id="myCanvas" width="350" height="150"></canvas>
```

Uvek treba navesti id atribut da bi canvas element bio povezan sa kodom iz JavaScript-a i širinu i visinu da bi se definisala veličina canvasa. Ukoliko se želi dodati okvir, to se radi pomoću *style* atributa:

```
<canvas id="myCanvas" width="350" height="150"
style="border: 1px solid red;">
</canvas>
```

Sva iscrtavanja po kanvasu se rade unutar JavaScript-a.

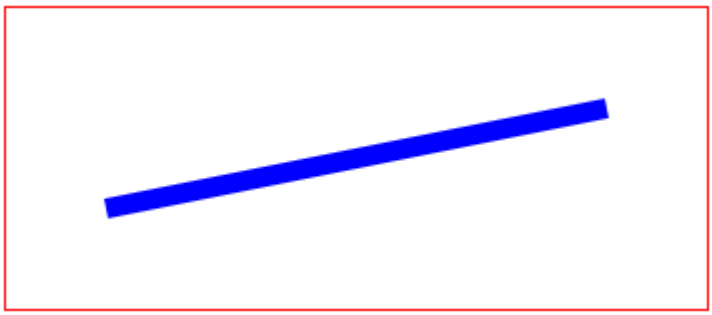
Canvas element ne podržava Internet Explorer 8 i ranije verzije.

6.1.1. Crtanje linije

```
<script >
var canvas=document.getElementById("myCanvas");
var context=canvas.getContext("2d");
context.lineWidth=10;
context.strokeStyle="blue";
context.moveTo(50, canvas.height-50);
context.lineTo(canvas.width-50,50);
context.stroke();
</script>
```

U primeru koda datom na početku ove sekcije je dat primer iscrtavanja jedne linije na canvasu. Prvo se pronalazi canvas element preko funkcije *getElementById.* Zatim se poziva metoda *getContext(″2d″)* sa obaveznim parametrom ″2d″ koji predstavlja objekat sa velikim brojem svojstava i metoda za crtanje putanja, krugova, teksta, itd. *lineWidth* definiše širinu linije od 10px,

strokeStyle definiše boju linije, *moveTo* definiše poziciju tačke od koje počinje iscrtavanje, *lineTo* crta liniju tako što definiše koordinate krajnje tačke linije, dok je *stroke* čini vidljivom.

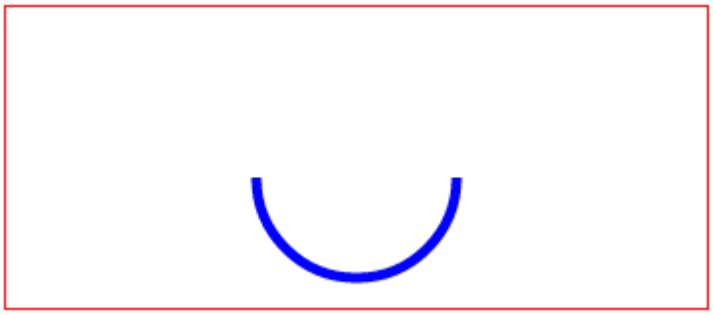


Slika 6.1.1. Linija na kanvasu

Na slici 6.1.1. je prikazan kanvas i linija iscrtana na njemu. Kanvas je oivičen crvenom linijom, širine 350px i visine 150px (u pitanju je canvas element definisan na kraju prethodne sekcije). Unutar njega se nalazi linija čije su osobine i pozicija definisane prethodnim kodom.

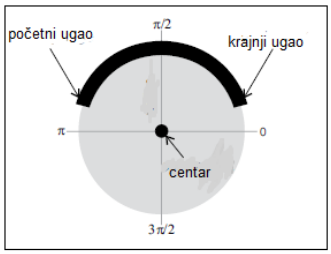
6.1.2. Crtanje luka

```
<script >
var canvas=document.getElementById("myCanvas");
var context=canvas.getContext("2d");
context.lineWidth=5;
context.strokeStyle="blue";
context.arc(canvas.width/2, canvas.height/2+10,50,0,1*Math.PI);
context.stroke();
</script>
```



Slika 6.1.2. Polukrug na kanvasu

Pomoću metoda *arc()* se definišu kružni oblici. Kao argumenti se uzimaju: pozicija centra po x koordinati odnosno u odnosu na širinu kanvasa, pozicija centra po y koordinati (visina kanvasa), poluprečnik, početni ugao, krajnji ugao. U datom kodu (što se može videti i sa slike 6.1.2.) polukrug se nalazi na sredini kanvasa (po širini) i pola visine stranice plus 10px. Poluprečnik je 50px, početni ugao 0, a krajnji π.

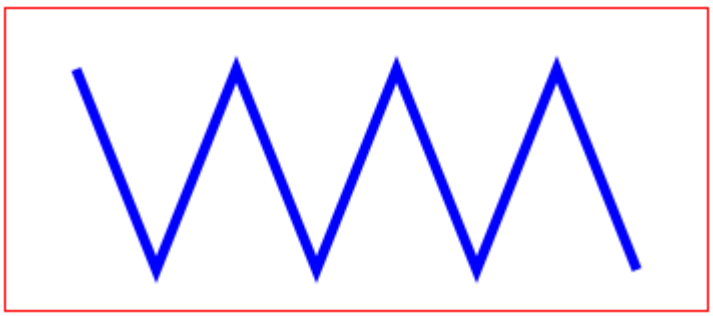


Slika 6.1.3. Argumenti arc metoda

Slika 6.1.3 daje prikaz definisanja uglova i smera iscrtavanja između tačaka definisanih početnim uglom i krajnjim uglom. Kao što se može videti, linija se povlači u smeru kazaljke na satu od tačke definisane početnim uglom do tačke definisane krajnjim uglom.

6.1.3. Crtanje izlomljene linije

```
<script >
var canvas=document.getElementById("myCanvas");
var context=canvas.getContext("2d");
var startX = 35;
var startY = 30;
var zigzagSpacing = 40;
context.lineWidth = 5;
context.strokeStyle = "blue";
context.beginPath();
context.moveTo(startX, startY);

for (var n = 0; n < 7; n++) {
var x = startX + ((n + 1) * zigzagSpacing);
var y;
if (n % 2 == 0) {
y = startY + 100;
}
else {
y = startY;
}
context.lineTo(x, y);
}
context.stroke();
</script>
```



Slika 6.1.4. Izlomljena linija na kanvasu

U datom kodu, promenljivom *startX* se definiše rastojanje izlomljene linije od početka kanvasa, po širini, dok se sa *startY* definiše isto to, samo po visini. Promenljivom *zigzagSpacing* se definiše širina između dve linije. *LineWidth* postavlja debljinu linije na 5px, dok *strokeStyle*

definiše boju linije i to je u ovom slučaju plava, što se može videti i sa slike 6.1.4.. Metod *beginPath()* označava crtanje linije, kao i da je kraj jedne neizlomljene linije početak naredne. Da se nije koristila ta metoda, pomoću *moveTo* bi se morala podesiti pozicija svakog segmenta izlomljene linije tako da se kraj jednog segmenta poklapa sa početkom narednog. Deo u *for* petlji crta sedam linija, dok *stroke()* omogućava da se one vide na kanvasu.

6.1.4. Tekst na kanvasu

```
<script >
var canvas=document.getElementById("myCanvas");
var context=canvas.getContext("2d");
context.font = "40pt Courier New";
context.fillStyle = "blue";
context.textAlign = "center";
context.textBaseline = "middle";
context.fillText("Tekst", canvas.width / 2, 75);
</script>
```

Slika 6.1.5. Tekst na kanvasu

Unutar kanvasa se može nalaziti i tekst. Pomoću svojstva *font* definiše se veličina i vrsta slova, *fillStyle* određuje koja će se boja slova koristiti. U ovom slučaju je to plava boja kao što se vidi i na slici 6.1.5.. *TextAlign* omogućava horizontalno poravnanje. U kodu je izabrano *center*, a pored toga može biti *left* ili *right*. *TextBaseline* služi za vertikalno poravnjanje i može biti *middle*, *top*, *bottom*, *hanging* i *alphabetic*. Sadržaj koji treba ispisati, kao i pozicija tog teksta se definiše unutar *fillText*.

6.1.5. 3D tekst sa senkom

```
<script >
var canvas=document.getElementById("myCanvas");
var context=canvas.getContext("2d");
context.font = "40pt Courier New";
context.fillStyle = "green";
context.textAlign = "center";
context.textBaseline = "middle";
ispisi3d(context, "3D Tekst", canvas.width / 2, 75, 7);
function ispisi3d(context, text, x, y, textDepth){
var n;
for (n = 0; n < textDepth; n++) {
context.fillText(text, x - n, y - n); }
context.fillStyle = "#5E97FF";
context.shadowColor = "green";
```

```
context.shadowBlur = 10;
context.shadowOffsetX = textDepth + 5;
context.shadowOffsetY = textDepth + 5;
context.fillText(text, x - n, y - n); }
</script>
```

Slika 6.1.6. 3D tekst na kanvasu

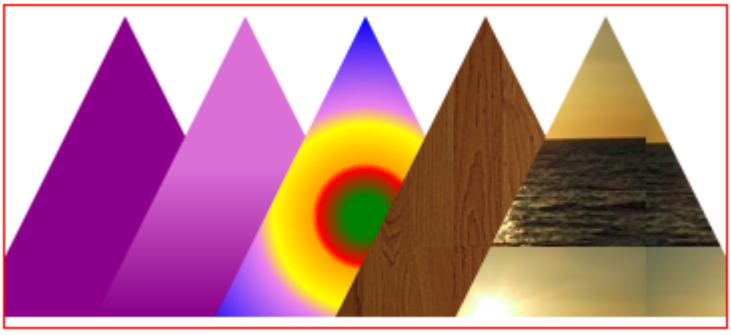
U poglavlju *6.1.4.* je objašnjeno kako se formira tekst. Da bi se formirao 3D tekst, potrebno je dodati funkciju (u ovom primeru *ispisi3d*) koja daje dubinu i senku tekstu. Kao argumenti te funkcije su uzeti *context*, tekst koji se ispisuje, koordinate x i y i dubina teksta. U svakom koraku *for* petlje se na već postojeći dodaje jedan sloj koji čini tekst dubljim. U ovom slučaju, to je zelena boja iza plavog teksta (slika 6.1.6.). Da bi se formirala senka iza 3D teksta, koriste se *shadowColor* (određuje boju senke), *shadowBlur* (određuje veličinu senke), *shadowOffsetX* (koliko je senka pomerena u desno u odnosu na tekst) i *shadowOffsetY* (pokazuje koliko je senka pomerena na dole u odnosu na tekst).

6.1.6. Različiti stilovi unutar trouglova

Unutrašnjost geometrijskih oblika je moguće popuniti na različite načine: bojom, različitim teksturama, moguće je umetnuti sliku,... Sledeći primer prikazuje 5 trouglova istih dimenzija iscrtanih na kanvasu i svaki od njih je ispunjen drugačije.

```
<script >
function drawTriangle(context, x, y, triangleWidth,
triangleHeight, fillStyle){
context.beginPath();
context.moveTo(x, y);
context.lineTo(x + triangleWidth / 2, y + triangleHeight);
context.lineTo(x - triangleWidth / 2, y + triangleHeight);
context.closePath();
context.fillStyle = fillStyle;
context.fill();
}
var canvas=document.getElementById("myCanvas");
var context=canvas.getContext("2d");
var grd;
var triangleWidth = 150;
var triangleHeight = 150;
var triangleY = canvas.height / 2 - triangleWidth / 2;
// početak definisanja prvog trougla
drawTriangle(context, canvas.width * 1 / 6, triangleY,
triangleWidth, triangleHeight, "#8B008B");
// kraj definisanja prvog trougla

// početak definisanja drugog trougla
```

```
grd = context.createLinearGradient(canvas.width * 2 / 6,
triangleY, canvas.width * 2 / 6, triangleY + triangleHeight);
grd.addColorStop(0.5, "#DA70D6");
grd.addColorStop(1, "#8B008B");
drawTriangle(context, canvas.width * 2 / 6, triangleY,
triangleWidth, triangleHeight, grd);
// kraj definisanja drugog trougla
// početak definisanja trećeg trougla
var centerX = (canvas.width * 3 / 6 +
(canvas.width * 3 / 6 - triangleWidth / 2) +
(canvas.width * 3 / 6 + triangleWidth / 2)) / 3;
var centerY = (triangleY +
(triangleY + triangleHeight) +
(triangleY + triangleHeight)) / 3;
grd = context.createRadialGradient(centerX, centerY, 10,
centerX, centerY, 100);
grd.addColorStop(0.15, "red");
grd.addColorStop(0.19, "orange");
grd.addColorStop(0.4, "yellow");
grd.addColorStop(0.003, "green");
grd.addColorStop(0.5, "violet");
grd.addColorStop(1, "blue");
drawTriangle(context, canvas.width * 3 / 6, triangleY,triangleWidth,
triangleHeight, grd);
// kraj definisanja trećeg trougla
// početak definisanja četvrtog trougla
var imageObj = new Image();
imageObj.onload = function(){
var pattern = context.createPattern(imageObj, "repeat");
drawTriangle(context, canvas.width * 4 / 6, triangleY,
triangleWidth, triangleHeight, pattern);
};
imageObj.src = "pattern.jpg";
// kraj definisanja četvrtog trougla
// početak definisanja petog trougla
var imag = new Image();
imag.onload = function(){
var picture = context.createPattern(imag, "repeat");
drawTriangle(context, canvas.width * 5 / 6, triangleY,
triangleWidth, triangleHeight, picture);
};
imag.src = "sun.jpg";
//kraj definisanja petog trougla
</script>
```



Slika 6.1.7. Trouglovi koji su ispunjeni na različite načine

Funkcija *drawTriangle* crta trouglove i započinje proces sa *beginPath().* Pozicija kursora koji crta menja se pomoću metode *moveTo(). LineTo()* definiše dve stranice trougla, dok se treća iscrtava metodom *closePath().* Prvi trougao sleva je ispunjen bojom, dok drugi trougao koristi linearnu gradaciju boje. Prelamanje boje se ostvaruje pomoću naredbe *createLinearGradient* koja za argumente uzima početnu i krajnju tačku:

var grd=context.createLinearGradient (startX, startY, endX, endY).

AddColorStop() definiše boju koja će se koristiti i njenu vrednost, odnosno poziciju. Može biti od 0 do 1, pri čemu 0 znači da se ta boja nalazi u vrhu trougla, dok 1 znači da je locirana u dnu. Treći trougao prikazuje radijalnu gradaciju koja se formira metodom *createRadialGradient()* gde se definiše početna tačka, početni poluprečnik, krajnja tačka i krajnji poluprečnik:

var grd=context.createRadialGradient (startX, startY, startRadius, endX, endY, endRadius).

Kod radijalne gradacije postoje dva imaginarna kruga. Prvi, koji je definisan početnim tačkama i drugi koji je definisan krajnjim tačkama. Boje su, kao i kod linearne gradacije definisane i pozicionirane sa *addColorStop().* U četvrtom i petom trouglu nalazi se slika, s tim što peti trougao ilustruje drvenu teksturu, a šesti neku sliku. Slika je umetnuta u trougao pomoću metode *createPattern()* koja zahteva naziv slike i opciju za ponavljanje:

var pattern=context.createPattern (imageObj, repeatOption).

Opcija za ponavljanje može imati vrednosti *repeat, repeat-x, repeat-y* i *no-repeat.* Podrazumevana vrednost je *repeat.*

6.2. SVG (Scalable Vector Graphics)

Koristi se za definisanje grafike bazirane na vektorima za veb. Definiše grafiku u XML formatu. SVG prikaz ne gubi kvalitet ukoliko se prikaz uvećava ili smanjuje.

U odnosu na ostale formate slika, SVG slike imaju određene prednosti: mogu da se kreiraju i menjaju u bilo kom tekstualnom editoru, mogu da se štampaju sa visokim kvalitetom u bilo kojoj rezoluciji, da se zumiraju bez degradacije i da se pretražuju, indeksiraju, kompresuju.

Za razliku od canvasa, SVG ne zavisi od rezolucije, podržava rukovanje događajima i nije pogodan za igrice.

Što se tiče podrške brauzera, SVG ne podržava Explorer 8 i ranije verzije.

SVG elementi se direktno uključuju u HTML stranicu i nalaze se unutar *body* taga.

6.2.1. Krug

```
<svg height="100" xmlns="http://www.w3.org/2000/svg">
      <circle  cx="50" cy="50" r="35" fill="violet" />
</svg>
```

Slika 6.2.1. Krug nacrtan korišćenjem SVG

Atributom *height* se određuje koji deo kruga će biti prikazan. U ovom slučaju je 100, odnosno čitav krug. *Circle* crta krug, *cx* i *cy* određuju poziciju u odnosu na širinu i visinu, respektivno. *R* je poluprečnik kruga, a *fill* određuje koja se boja koristi.

6.2.2. Pravougaonik

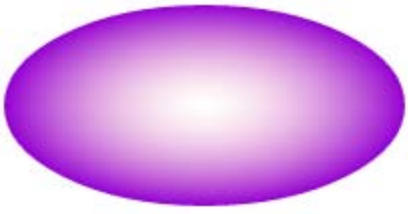
```
<svg height="100" xmlns="http://www.w3.org/2000/svg">
    <rect width="200" height="70" fill="violet" />
</svg>
```

Slika 6.2.2. Pravougaonik nacrtan korišćenjem SVG

Rect crta pravougaonik, *width* i *height* definišu širinu i visinu pravougaonika, a *fill* određuje boju. Na slici 6.2.2. je prikazana realizacija koda.

6.2.3. Elipsa

```
<svg  height="100" xmlns="http://www.w3.org/2000/svg">
   <defs>
      <radialGradient id="gradient" cx="50%" cy="50%" r="50%"
      fx="50%" fy="50%">
      <stop offset="0%" style="stop-color:rgb(255,193,193);
      stop-opacity:0"/>
      <stop offset="100%" style="stop-color:rgb(148,0,211);
      stop-opacity:1"/>
      </radialGradient>
   </defs>
   <ellipse cx="100" cy="50" rx="100" ry="50"
      style="fill:url(#gradient)" />
</svg>
```



Slika 6.2.3. SVG gradijentna elipsa

RadialGradient određuje radijalni gradijent, *cx* i *cy* definišu koliko će unutrašnja (svetlija) elipsa biti pomerena ulevo, odnosno na dole. Postavljanjem tih vrednosti na 50%, ona se nalazi u sredini. Povećavanjem *r* svetlija elipsa postaje dominantnija i prekriva tamniju, dok *fx* i *fy* pomeraju tamniju elipsu ulevo, odnosno udesno. Postavljanjem *offset*-a na 0% svetlo roza boja se nalazi u centru, a postavljanjem na 100% ljubičasta se nalazi pri krajevima, dok deo između predstavlja gradaciju tih boja. *Ellipse* crta elipsu.

7. APLIKACIJE

HTML5 je učinio stvaranje aplikacija lakšim. Buduće veb aplikacije će se odlikovati sledećim karakteristikama:

- Lokalno skladištenje podataka
- Aplikacije sa keširanjem podataka
- JavaScript radnici (workers)
- Server – sent događaji
- Geografska lokacija

7.1. Lokalno skladištenje podataka

Ranije je skladištenje podataka bilo moguće uz pomoć kolačića, međutim lokalno skladištenje je mnogo sigurniji i brži način. Moguće je skladištiti i velike količine podataka bez uticaja na performanse veb sajta. Dobre strane korišćenja ovog skladištenja ogledaju se u tome što korisnik može da koristi aplikaciju i kada je oflajn. Podaci se čuvaju u parovima ključ – vrednost i veb strana može pristupiti podacima koji su uskladišteni od strane brauzera. Za razliku od kolačića, količina podataka koji se skladište je mnogo veća (najmanje 5 MB) i informacije se nikada ne prosleđuju serveru.

Postoje dva nova objekta za skladištenje podataka na klijentu: *localStorage* i *sessionStorage*.

Ovakvo skladištenje podataka ne podržava Explorer 7 i ranije verzije, ali pre korišćenja je ipak potrebno proveriti da li brauzer podržava *localStorage* i *sessionStorage:*

```
<script>
   if(typeof(Storage)!=="undefined")
  {
  // Brauzer podržava lokalno skladištenje podataka }
  else
  {// Brauzer ne podržava lokalno skladištenje podataka}
</script>
```

Operator *typeof* daje objašnjenje onoga što je u zagradi. Na primer, da je pisalo: *alert (typeof (5))*, dobilo bi se obaveštenje da je to broj. U prethodnom kodu, postavlja se uslov ako je skladištenje definisano onda brauzer to podržava, u suprotnom ne podržava.

7.1.1. LocalStorage objekat

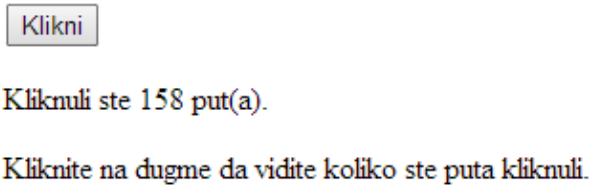
LocalStorage objekat skladišti podatke bez datuma isteka i podaci neće biti izbrisani kada se zatvori brauzer. Parovi ključ/vrednost se uvek skladište kao string i treba voditi računa o tome da se konvertuju u neki drugi tip kada je to potrebno.

```
if (typeof(Storage) != "undefined")
  {localStorage.setItem("ime", "Marko");
    localStorage.setItem("prezime", "Markovic");
 document.getElementById("rezultat").innerHTML=localStorage.getItem("prezime");}
else
  {document.getElementById("rezultat").innerHTML="Brauzer ne podržava lokalno
skladištenje";}
```

U ovom primeru su pomoću *setItem* kreirana dva *localStorage* para ključ/vrednost, jedan ključ je ime čija je vrednost Marko, a drugi prezime čija je vrednost Markovic. *getItem* navodi koji ključ se bira i u brauzeru se ispisuje njegova vrednost, u ovom primeru Markovic.

Sledeći primer pokazuje koliko je puta korisnik kliknuo na dugme. Pre toga, string se mora konvertovati u broj naredbom *localStorage.clickcount=Number(localStorage.clickcount)+1;* gde ujedno i svaki put kada korisnik klikne na dugme, uvećava taj broj za jedan.

```
<!DOCTYPE html>
<html>
  <head>
  <meta charset="utf-8">
    <title>Storage</title>
      <script>
function izbroj()
{
if(typeof(Storage)!=="undefined")
  {
  if (localStorage.clickcount)
    {
    localStorage.clickcount=Number(localStorage.clickcount)+1;
    }
  else
    {
    localStorage.clickcount=1;
    }
  document.getElementById("rezultat").innerHTML="Kliknuli ste " +
localStorage.clickcount + " put(a)."; }
else
  {
  document.getElementById("rezultat").innerHTML="Brauzer ne podržava lokalno
skladištenje";
  }}
</script>
  </head>

  <body>
   <p><button onclick="izbroj()" type="button">Klikni</button></p>
<div id="rezultat"></div>
<p>Kliknite na dugme da vidite koliko ste puta kliknuli.</p>
<p>Brojač će nastaviti da broji kada se ponovo otvori brauzer.</p>
  </body>
</html>
```



Slika 7.1.1. Izgled ekrana kada se pokrene prethodni kod

S obzirom da je korišćen *localStorage*, svaki put kada se zatvori brauzer i ponovo otvori i pokrene kod, brojač se ne resetuje, nego prikazuje vrednost uvećanu za jedan u odnosu na prethodnu vrednost što se može videti i na slici 7.1.1. Da bi se izbrisali podaci i brojač krenuo da broji od jedinice svaki put kada se kod pokrene, u prethodnom kodu iznad *</script>* treba dodati deo: *localStorage.clear();*. Sačuvani podaci se ne mogu prebaciti na drugi računar i koristiti na isti način kao na prvom računaru.

7.1.2. SessionStorage objekat

Za razliku od *localStorage*, *sessionStorage* skladišti podatke samo za jednu sesiju. Podaci se brišu kada korisnik zatvori brauzer.

Da se u poslednjem primeru u odeljku 7.1.1. umesto *localStorage* koristilo *sessionStorage*, svaki put kada se aktivira kod brojač bi se resetovao i počeo da broji od jedinice, a ne od vrednosti koja je bila u poslednjoj sesiji.

7.2. Aplikacije sa keširanjem podataka

Od sve većeg značaja je dostupnost aplikacija kada je korisnik oflajn. Prednosti appCache se ogledaju u sledećem:

- Offline pretraživanje – korisnici mogu koristiti aplikaciju kada su oflajn
- Brzina – keširani podaci se brže učitavaju
- Smanjeno opterećenje servera – brauzer će jedino da skida promenjene/ažurirane podatke sa servera

Da bi se omogućilo keširanje, potrebno je uključiti *manifest* atribut unutar html taga:

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

Svaka stranica na kojoj je uključen ovaj atribut, biće keširana kada je korisnik poseti. Preporučena ekstenzija manifest fajla je *.appcache*. To je jednostavan tekstualan fajl koji će pokazati brauzeru šta da kešira. Manifest fajl se sastoji iz tri dela:

- CACHE MANIFEST – fajlovi navedeni ispod ovog zaglavlja biće keširani kada se prvi put daunlouduju.

```
CACHE MANIFEST
/theme.css
/logo.gif
/main.js
```

U ovom primeru su navedena tri resursa: css fajl, gif slika i JavaScript. Kada se manifest fajl učita, brauzer daunlouduje navedene resurse iz root direktorijuma veb sajta koji će uvek biti dostupni korisniku čak i kada nije povezan na Internet.

- NETWORK – fajlovi navedeni ispod ovog zaglavlja zahtevaju konekciju na server i nikad se neće keširati.

```
NETWORK:
login.asp
```

Fajl *login.asp* se neće keširati i neće biti dostupan korisniku kada je oflajn.Ukoliko se ispod NETWORK dela nalazi zvezdica, to znači da resursi koji nisu navedeni pod CACHE MANIFEST zahtevaju internet konekciju.

```
NETWORK:
*
```

- FALLBACK – fajlovi navedeni ispod ovog zaglavlja predstavljaju rezervne fajlove kojima će se pristupiti ukoliko je stranica nedostupna.

```
FALLBACK:
/html/ /offline.html
```

Ako se ne ostvari internet konekcija, umesto fajlova u html direktorijumu će biti prikazana stranica offline.html.

Ukoliko je jednom izvršeno keširanje, ono može biti prekinuto ako se desi da je:

- korisnik obrisao keš brauzera
- manifest fajl modifikovan
- keš programski ažuriran.

7.3. JavaScript radnici (workers)

Kada se izvršavaju skripte u html strani, stranica postaje blokirana sve dok se skripte ne izvrše. Web worker je JavaScript skripta definisana od strane W3C i WHATWG koja se izvršava u pozadini, nezavisno od ostalih skripti i ne utiče na performanse strane. Korisnik može da nastavi da radi na stranici bilo šta, dok se web worker izvršava nezavisno. Web worker ne podržava Explorer 10 i ranije verzije.

Neka se formira neka skripta koja se čuva u eksternom fajlu i koja se zove *workers.js*. Sledeće što treba uraditi je da se pozove sa html stranice. Sledeći kod ispituje da li worker postoji i ako ne postoji formira se novi objekat i poziva se fajl *workers.js.*

```
if(typeof(w)=="undefined")
  {
  w=new Worker("workers.js");
}
```

Da bi se prekinuo web worker poziva se metoda *terminate():*

```
w.terminate();.
```

7.4. Server – Sent događaji

Server – sent predstavlja događaj kada veb strana automatski prima notifikacije poslate sa servera. Ovo je bilo moguće i ranije, ali je veb strana morala da pita da li ima dostupnih promena. Primer gde se koriste server – sent događaji su fejsbuk, tviter, sprotski rezultati, vesti, promene cena na berzi,..

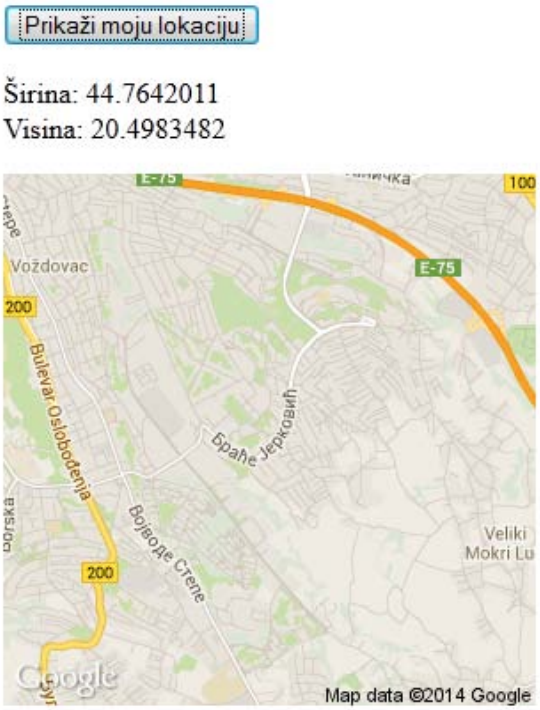
7.5. Geografska lokacija

HTML5 omogućava određivanje lokacije korisnika. S obzirom da to narušava korisnikovu privatnost, određivanje je moguće tek nakon što korisnik dozvoli. Geolokacija je mnogo tačnija za uređaje koji imaju GPS.

Geolokacija nije moguća na Exploreru 8 i ranijim verzijama.

```
<!DOCTYPE html>
<html>
  <head>
  <meta charset="utf-8">
    <script type="text/javascript"
      src="https://maps.googleapis.com/maps/api/js?sensor=false">
    </script>
    <script type="text/javascript">
       function pronadjiLok() {
  var output = document.getElementById("out");

  if (!navigator.geolocation){
    output.innerHTML = "<p>Brauzer ne podržava geolokaciju</p>";
    return;
  }

  function uspeh(position) {
    var latitude  = position.coords.latitude;
    var longitude = position.coords.longitude;

    output.innerHTML = '<p>Širina: ' + latitude + ' <br>Visina: ' + longitude ;

    var img = new Image();
    img.src = "http://maps.googleapis.com/maps/api/staticmap?center=" + latitude
+ "," + longitude + "&zoom=13&size=300x300&sensor=false";

    output.appendChild(img);
  };

  function greska() {
    output.innerHTML = "Ne može se pronaći lokacija!";
  };

  output.innerHTML = "<p>Traženje lokacije...</p>";

  navigator.geolocation.getCurrentPosition(uspeh, greska);
}
    </script>
  </head>
  <body>
    <p><button onclick="pronadjiLok()">Prikaži moju lokaciju</button></p>
```

```
<div id="out"></div>
  </body>
</html>
```



Slika 7.5.1. Prikaz lokacije

Pre nego što se pristupi određivanju lokacije, potrebno je proveriti da li brauzer podržava geolokaciju. To se proverava sa *if (!navigator.location).* Ukoliko je uslov ispunjen, znači da ne podržava i ispisuje se poruka *„Brauzer ne podržava geolokaciju“*. Funkcija *uspeh* pronalazi koordinate i daje adresu mape na kojoj se prikazuje tačna lokacija. Metod *appendChild* prikazuje sliku, odnosno mapu na brauzeru. Funkcija *greska* se aktivira ako se ne mogu pronaći koordinate. Sa metodom *getCurrentPosition* dobija se trenutna pozicija, ima dva argumenta: funkcije *uspeh* i *greška*. Ukoliko se pronađe, aktivira se funkcija *uspeh*, u suprotnom *greška*. Na slici 7.5.1. su prikazane koordinate trenutne lokacije i pozicija na mapi.

8. ZAKLJUČAK

Prethodna verzija html-a, HTML4, bila je veoma statična i uvođenjem novog standarda, HTML5, uvodi se dinamičnost koja omogućava bolju i napredniju verziju korisničkog interfejsa. U kombinaciji sa CSS-om omogućava kreiranje veoma interaktivnih i dinamičnih veb sajtova bez potrebe korišćenja dodataka.

Podržan je i od strane mobilnih uređaja, tako da je sada veoma jednostavno razviti veb sajt ili aplikaciju u HTML5 koja će se podjednako dobro prikazivati na desktop ili laptop računaru, ali isto tako i na bilo kom mobilnom uređaju.

Novi input elementi i atributi omogućavaju bolju kontrolu unosa podataka i njihovu validaciju bez potrebe korišćenja JavaScript-a. Canvas element je našao svoju primenu za crtanje vektorske grafike, različitih efekata i animacija. Web storage mehanizam omogućuje jednostavniji, brži i sigurniji način za skladištenje podataka.

Ono što je mana HTML5 je da nisu svi elementi podržani od strane svih brauzera, pa u zavisnosti od toga u kom se brauzeru otvara stranica, postoji mogućnost da sve funkcionalnosti HTML5 neće biti prikazane.

LITERATURA

[1] J. Keith, „HTML5 for Web Designers“, 2010.

[2] O’Reilly, „HTML5, Up and Running“, 2010.

[3] F. Wempen, „HTML5 Step by Step“, 2011.

[4] C. Grannell, V. Sumner, D. Synodinos, „HTML5 and CSS3 Web Design“, 2012.

[5] http://www.w3schools.com/